Regular Paper

Improvement of Neural Reverse Dictionary by Using Cascade Forward Neural Network

Yuya Morinaga[1,a)] Kazunori Yamaguchi[1,b)]

Received: February 21, 2020, Accepted: August 7, 2020

Abstract: A reverse dictionary maps a description to the word specified by the description. The neural reverse dictionary (NRD) learns to map word embeddings for an input definition into an embedding of the word defined by the definition using neural networks. Such a function encodes phrasal semantics and bridges the gap between them and lexical semantics. However, previous NRD has a limitation in accuracy due to its insufficient capacity. To solve this problem, we used novel combinations of neural networks, which are effective in neural machine translation and image processing, with sufficient capacities. We found that, an LSTM output adjustment by using a multi-layer fully connected network with bypass structures (CFNN) was more effective for reverse dictionary tasks than using more complicated LSTM. BiLSTM+CFNN was comparable to the commercial system OneLook Reverse Dictionary in some metrics, and noised biLSTM+CFNN which we tuned by a noising data augmentation outperformed OneLook Reverse Dictionary in almost all metrics. We also examined the reasons for the success of biLSTM+CFNN and revealed that a bypass structure of the CFNN and balance in the capacity of LSTM and the CFNN contribute to the improved performance of the NRD.

Keywords: neural reverse dictionary (NRD), long short-term memory (LSTM), cascade forward neural network (CFNN), convolutional neural network (CNN), data augmentation, noising input

1. Introduction

A dictionary maps a word to its definition, while a reverse dictionary maps a description to the word specified by the description. Direct reverse dictionary applications include the tip-of-the-tongue problem [1] and cross word problem [2]. A reverse dictionary can map an input sentence to its meaning. Such a function encodes phrasal semantics and bridges the gap between them and lexical semantics [2].

One of the difficulties in developing a reverse dictionary is that we cannot exhaustively enumerate descriptions for a word. For example, the definition of the word "brother" in WordNet [3] is "a male with the same parents as someone else", but a reverse dictionary should also be able to map the description of "son of my parents" to "brother".

To achieve this, a reverse dictionary should be able to calculate the similarity between unseen inputs and candidate words. Several studies have proposed models for reverse dictionaries. One study [4] used a lexical database, and others [5], [6], [7], [8] used Wordnet. These studies proposed hand-engineered features of sentence and heuristics to search words from an input sentence. Another study [2] used word embeddings as neural language models [9] to create a reverse dictionary (we call it the Neural Reverse Dictionary/NRD). That study also used machine learning.

The NRD [2] converts the word embeddings of word2vec [10] for an input description into a vector using a linear transformation or recurrent neural network. This reverse dictionary was claimed by Ref. [2] to perform similar to that of OneLook Reverse Dictionary, which is a commercial specialty reverse dictionary system indexed with 1,061 dictionaries, at the moment of its evaluation. We improved the NRD by filtering candidate words by the estimated category of the target word [11].

The NRD has a substantial advantage in that it can generate reverse dictionaries from dictionaries without human intervention. However, in 2018, the performance of the NRD of Refs. [2], [11] was far inferior to that of OneLook Reverse Dictionary.

We investigated the cause of the low performance of the NRD and found that its capacity is insufficient; thus, with controlled increase in model capacity, we can improve its performance. We tested several neural network models for the NRD. The combination of bidirectional long short-term memory and the cascade forward neural network (biLSTM+CFNN) was the most accurate and that the NRD even outperformed the OneLook Reverse Dictionary in some metrics. We could improve the performance of the NRD from limited resources to match the performance of OneLook Reverse Dictionary, which can access enormous resources and can be tuned by experts.

We also examined reasons for the success of using biLSTM+CFNN. We then revealed the following three facts: 1) LSTM structure's output adjustment by a multi-layer fully connected network can be effective for reverse dictionary tasks, 2) the bypass structure in a multi-layer fully connected network is essential to properly learn the NRD, and 3) the balance in capacity

[1] Department of General Systems Studies, The University of Tokyo, Meguro, Tokyo 153–8902, Japan
[a)] morinaga-yuya@g.ecc.u-tokyo.ac.jp
[b)] yamaguch@g.ecc.u-tokyo.ac.jp

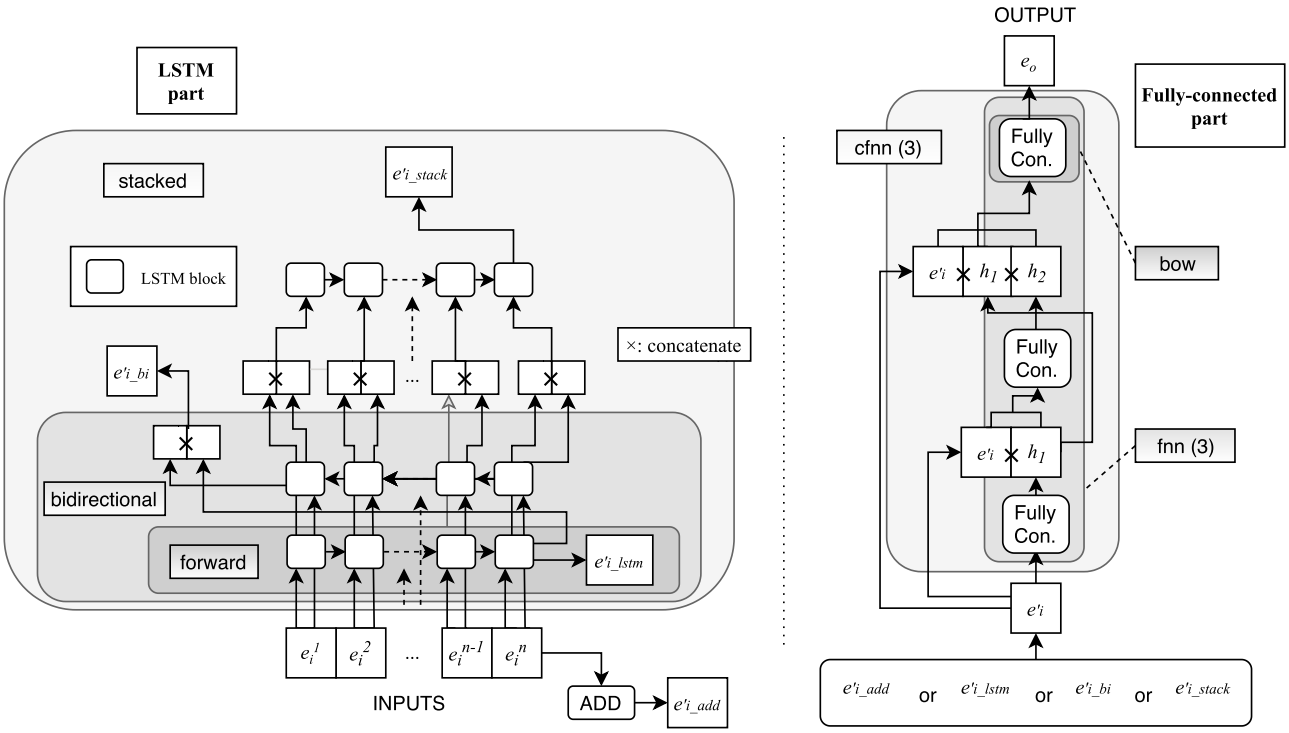


Fig. 1 Network architectures. (left) LSTM part: ADD, LSTM, biLSTM, and stacked-LSTM are overlaid. (right) Fully connected part: bow, FNN, and CFNN are overlaid. See Table 1 for combinations.

of LSTM and the multi-layer fully connected network improve the performance of the NRD. BiLSTM+CFNN satisfies all these conditions.

We also tuned our best combination, i.e., biLSTM+CFNN using two techniques that are effective for reverse dictionary or other natural language processing (NLP) tasks. The first is category inference of the target word by using a convolutional neural network (CNN), which we previously showed to be effective for reverse dictionary tasks [11]. The second is data augmentation by noising inputs, which was shown to be beneficial for some NLP tasks, e.g., autoencoder [12], [13] and back translation [14]. Both techniques boosted the performance of biLSTM+CFNN. We also attempted fine-tuning the pre-trained NLP model BERT [15] for the NRD and revealed that this approach did not work well under limited computational resources for training.

The rest of this paper is organized as follows. We explain the NRD and discuss an empirical study on its performance characteristics in Section 2. In Section 3, we compare our new model combinations having improved capacities over the previous models. In Section 4, we explain the experiments we conducted and present the results. We present our best model combination, i.e., biLSTM+CFNN, in Section 5. We give a conclusion and mention future work in Section 6.

2. Previous NRD Models and Their Limitations

We explain the previous NRD models and examine their limitations in Sections 2.1 and 2.2, respectively.

2.1 Previous NRD

The NRD outputs words sorted by rank determined by the cosine similarity between the vector calculated from an input description and the vectors of candidate words. The NRD is based on a neural language model [9]. Word embeddings may be learned independently from or learned simultaneously with NRD parameters. Hill et al. [2] reported that there was almost no difference in the results obtained in either case. Hence, we used a pre-trained word2vec for word embeddings. Dataset and preprocessing for pre-training word2vec are the same as those in our previous study [11]. Training conditions are using continuous bag-of-words and hierarichical softmax, a window size of 5, sampling rate of 10^{-5}, min-count threshold of 10, and 10 iterations.

Table 1 Combination of the network architectures. See Fig. 1 for the network architectures.

	LSTM part	Fully-connected part
BOW	none (ADD)	bow
LSTM	forward	bow
FNN	none (ADD)	fnn
CFNN	none (ADD)	cfnn
biLSTM	bidirectional	bow
stacked-LSTM	stacked	bow
biLSTM + CFNN	bidirectional	cfnn
stacked-LSTM + CFNN	stacked	cfnn

Next, we explain the neural networks of the NRD used in previous studies [2], [11]. We show these models' architectures in **Table 1** and **Fig. 1**.

2.1.1 word2vec Add (ADD)

Hill et al. [2] used the sum of the word embeddings of words in a description as the approximation of its meaning. We call such a model the *word2vec add model* (ADD). No learning is needed with the ADD when pre-trained word embeddings are used. In our experiments, we omitted word embeddings of stop words in the summation, as in our previous study [11]. This omission slightly improves the accuracy of the ADD. We used this improved ADD as the baseline in our study.

2.1.2 Bag of Words (BOW)

The bag of words distributed representation model (BOW) [2] outputs a linearly transformed summation of the vectors of input words. The order of words is ignored with this model. The cosine distance is used as a cost function.

2.1.3 Long Short-Term Memory (LSTM)

Hill et al. [2] also used long short-term memory [16] (LSTM), which takes the word order into account. The activation function of the LSTM output is tanh and that of LSTM gate (recurrent activation) is hard_sigmoid [*1].

2.1.4 CNN based Category Inference NRD

In our previous study [11], we introduced a convolutional neural network (CNN)-based category-inference into the NRD, which combines the previous vector generator with inference on the category of the target word. We trained this CNN by a word category and the description of the word. BOW or LSTM searches the target word in the categories predicted with the trained CNN.

2.2 Empirical Study on NRD with Previous Models

In this section, we discuss the possible limitations of the previous models for the NRD.

2.2.1 Training Data

We used about 937k word and definition pairs obtained from Wordnik API [*2]. This pairs include about 260k kinds of defined words (approximately 4 definitions per word). We used 1% of the data as the validation data to determine early stopping and the remaining 99% for actual training. Selection of this 1% substantially affected the performance of the NRD, but detailed examination is beyond the scope of this paper.

For the CNN training, we used about 120k lexname and definition pairs in Wordnet. We used 10% of the training data for validation.

This data were already tokenized, and we did not pre-process (lemmatization, stemming, and so on) them further.

2.2.2 Test Data and Evaluation Metrics

There are two possible evaluation metrics: the precision of the unseen dictionary data (word and description pairs) not used in the training, and the precision of the user description (target word and its description pairs). From the former, generalization performance can be measured using the data having the same characteristics as the training data; thus, we could determine whether the training was successful. This is appropriate for model validation in the learning phase, so we used unseen dictionary data only for the NRD model validation (see also Section 2.2.1), but not for the NRD model test. From the latter, the performance in the intended use for an actual reverse dictionary can be measured. This is appropriate for the NRD model test. We employed the user description as test data for the NRD evaluation.

We searched for a word from a description using the NRD to evaluate and determine the precision from the rank of the target word in the candidate word list generated using the NRD. We used the metrics accuracy @ 1/10/100 [*3], which are the rates at which the target word is included in the candidate word list up to the 1st/10th/100th in rank, and the median of the ranks. We used all the words defined in Wordnet as target words vocabulary because the category information is only given to words in Wordnet. This was not a restriction because all the target words in the user descriptions were in Wordnet.

Table 2 Performance of NRD with previous models and OneLook Reverse Dictionary.

Model	accuracy@1/10/100	median
ADD	0.02 / 0.25 / 0.60	52
BOW	0.03 / 0.30 / 0.67	38
CNN + BOW	0.06 / 0.32 / 0.69	31
LSTM	0.10 / 0.30 / 0.65	37
CNN + LSTM	0.12 / 0.33 / 0.67	36
OneLook	0.34 / 0.55 / 0.76	6

Table 3 Comparison of construction conditions of the NRD and OneLook Reverse Dictionary.

	NRD	OneLook
resources	(at most five) freely available dictionaries	almost 1,000 dictionaries
available user data	none	user search logs
openness	most codes are publicly available	architecture and codes are not publicly available

2.2.3 Evaluation and Findings

We re-implemented the previous NRD models [*4] and evaluated their performances. The results are listed in **Table 2**, where CNN+BOW and CNN+LSTM mean the combination of CNN category inferences mentioned in Section 2.1.4 with these models.

The models with learning, BOW, CNN+BOW, LSTM, and CNN+LSTM, performed better than that without learning, i.e., the ADD. Thus, neural network transformations are effective in reverse dictionary tasks. However, the NRD with CNN + LSTM, which is the best of the previous NRD models, performed much worse than OneLook Reverse Dictionary.

There are several differences in the amount of lexical resources, availability of user data at their constructions [*5] and openness of the system between the NRD with the previous models and OneLook Reverse Dictionary, as shown in **Table 3**.

From these differences, we consider that there are two possible reasons for the poorer performance with the previous NRD models.

(1) The training data are insufficient.

(2) The models do not have sufficient capacity.

First, the relationship between the training data size and accuracy of LSTM is shown in **Fig. 2**, where 'val' means the validation by using the dictionary data and 'user' means the test by using the user descriptions [*6]. Similar results were obtained for BOW.

[*1] Linear approximation of sigmoid.

[*2] http://developer.wordnik.com

[*3] accuracy@1 is new for this paper.

[*4] The code will be available at https://gitlab.com/morinagy/neuralreversedictionary_morinaga.

[*5] The NRD cannot access to any user data during training phase under our experimental conditions.

[*6] In Fig. 2, we see that the 'val' accuracies are lower than 'user' accuracies, which is contradictory to the intuition that it is more difficult to predict for out-of-domain data than in-domain data. This counter-intuitive result comes from differences in targets contained each data. We picked (a) 200 definitions from training data of which targets are same to the user descriptions, and (b) random 200 definitions from the training data. The LSTM accuracy@10 for (a) was 0.42, while that for (b) was only 0.22. Dictionaries define many uncommon words, and accuracies for them are relatively low.

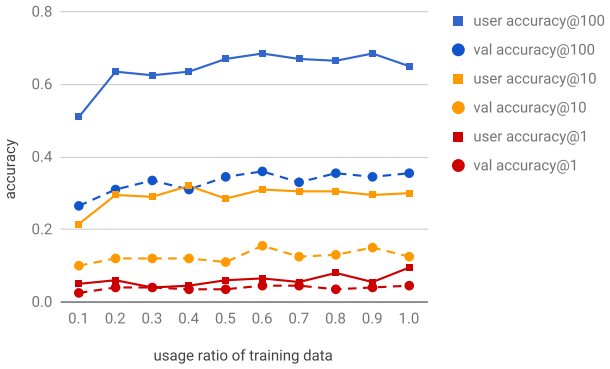


Fig. 2 Accuracy vs. training data size (LSTM).

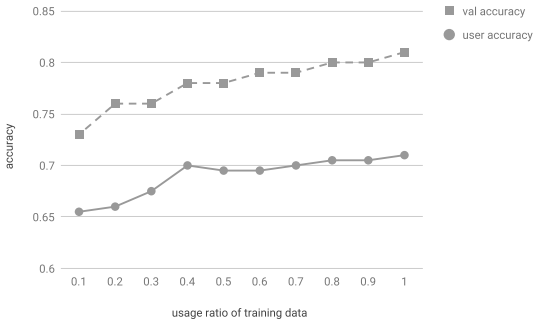


Fig. 3 Accuracy vs. training data size (CNN category classifier).

We see that validation accuracies are saturated with the amount of training data in Fig. 2. This means that the training data are sufficient for learning with the previous NRD models. This refutes possible reason 1.

Possible reason 2 remains after the first reason was refuted.

Next, we studied the relationship between the amount of training data and accuracy of CNN category inference, as shown in **Fig. 3**. With more training data, the classification accuracies for both validation data and the user description increased. This suggests that the poor performance of category inference is due to possible reason 1.

We focused on possible reason 2 and improved the NRD models in this paper.

3. Proposed Neural Network Models for NRD

In this section, we introduce new types of models for reverse dictionary in addition to the those introduced in Section 2.1. We show each of our model's architecture in Table 1 and Fig. 1, along with the previous models.

3.1 Feed-forward Neural Network (FNN)

It is natural to extend BOW (a single layer perceptron) to a multi-layered perceptron when more capacity is required. We used a feed-forward neural network (FNN) consisting of multiple layers of densely connected networks. Our FNN consists of 5 layers. We used a rectified linear unit (ReLU) as an activation function.

3.2 Cascade Forward Neural Network (CFNN)

We also tested the CFNN, which is an FNN augmented with additional connections between non-adjacent layers. These additional connections are based on the cascade-correlation learning architecture [17]. We call these additional connections the *bypass structure*. The CFNN is similar to widely used densely connected CNNs [18], but the CFNN has no pooling nor softmax layers because its output is used as an approximation of the target vector. Our CFNN consists of 5 layers. We also used a ReLU as an activation function.

3.3 Bidirectional LSTM (biLSTM)

Because the function of the NRD resembles that of machine translation, we tested neural networks used in neural machine translation. Bidirectional LSTM (biLSTM) [19] is a combination of two LSTMs using input in the forward and backward directions. We used hard_sigmoid as a recurrent activation function and tanh as another activation functions.

3.4 BiLSTM + CFNN

The combination biLSTM+CFNN uses biLSTM for the first layer, and the concatenation of the outputs of forward LSTM and backward LSTM is fed into the second layer of the CFNN. BiLSTM is intended to extract the context vector of the input description and the CFNN is intended to transform the context vector into the target vector. We focused on biLSTM + CFNN in our experiment. The order of the input sequence is captured using biLSTM and complex vector adjustments is carried out using the CFNN. This combination of biLSTM and CFNN worked the best in our experiment.

3.5 Stacked-LSTM

Stacked-LSTM is a network consisting of multiple LSTM structures with stacked LSTM blocks as in Fig. 1. We used tanh as an activation function.

3.6 Stacked-LSTM + CFNN

Stacked-LSTM + CFNN is a combination of stacked-LSTM and the CFNN. We used stacked-LSTM because it can extract better context than biLSTM. Other models, such as densely connected-LSTM [20], were not tested because their computational cost is too high.

4. Experiments and Results

In this section, we explain the experimental conditions (Section 4.1), present the experimental results (Section 4.2), and provide further discussion (Section 4.3).

4.1 Experimental Condition

We used two types of datasets for training. The first is the dataset introduced in Section 2.2.1, which is the same as the training dataset we previously used [11], and is 937k pairs. The Second, is the first dataset plus the Oxford Dictionary [*7] and has 1,181k pairs. Conditions for the data were the same as those mentioned in Section 2.2.1. The loss function was cosine distance, optimizer was Adam [21], learning rate was 0.001, and batch size was 500. The total numbers of trainable parameters for the models are summarized in **Table 4**.

[*7] https://www.oxforddictionaries.com

Table 4 Trainable parameters and convergence time for each model (* indicates the previous models).

MODEL	trainable parameter	convergence time (hour(s))
* BOW	250,000	0.03
* LSTM	2,002,000	1
FNN	5,000,000	0.05
CFNN	15,250,000	0.05
biLSTM	13,008,000	3
stacked-LSTM	32,516,000	9
biLSTM + CFNN	78,008,000	4
stacked-LSTM + CFNN	87,516,000	10

Table 5 Accuracy @1/10/100 (denoted as @1, @10, and @100 in column titles) and rank median of each model for test dataset. "*l* / *r*" means score of *l* of model trained on dataset (a) and score *r* of model trained on dataset (b). Scores averaged over four runs are shown except for NRD with previous models (*) and OneLook Reverse Dictionary.

MODEL	@1	@10	@100	median
* LSTM	0.07 / 0.07	0.32 / 0.34	0.68 / 0.72	33.0 / 26.0
FNN	0.16 / 0.12	0.41 / 0.38	0.68 / 0.68	19.0 / 28.0
CFNN	0.10 / 0.15	0.41 / 0.43	0.74 / 0.71	18.5 / 17.5
biLSTM	0.14 / 0.18	0.43 / 0.52	0.74 / 0.77	15.5 / 9.8
biLSTM + CFNN	0.20 / **0.24**	0.52 / **0.58**	0.77 / **0.82**	9.3 / **6.3**
stacked-LSTM	0.19 / 0.20	0.50 / 0.52	0.78 / 0.79	11.3 / 9.0
stacked-LSTM + CFNN	0.21 / 0.22	0.51 / 0.52	0.76 / 0.77	9.8 / 8.5
OneLook (2018)	**0.34**	0.55	0.76	**6.0**

4.2 Experimental Results

The experimental results are listed in **Table 5**. Considering randomness, the averaged scores over four runs are shown. All of our models performed better than the previous models, for both training datasets (a) and (b). Particularly, biLSTM + CFNN substantially outperformed the previous models. The NRD with this model trained on dataset (b) even outperformed commercial OneLook Reverse Dictionary in accuracy@10/100.

Almost all the NRD models that contain LSTM structures were more accurate when being trained on dataset (b) than on (a). The difference between datasets (a) and (b) is only the Oxford Dictionary. Dataset (a) is a large dataset (937k pairs) that contains only old free dictionaries, while dataset (b) contains the Oxford Dictionary as well, which is not so large (1,181k−937k = 244k pairs) compared to dataset (a) but still now maintained commercial dictionary. LSTM structures take into account word orders, so those performance gaps may be due to dictionary-data quality rather than from its quantity. Our NRD models were far more accurate than the previous ones under the same training data conditions (dataset (a)) to that of a previous study, and performed better on dataset (b). Below are only the results of the NRD models trained on dataset (b).

4.3 Additional Experiments

4.3.1 Target Category Inference by CNN for NRD

We previously showed that the category inference of the target word by using a CNN can be used to improve the NRD's performance [11], and this category inference can be used to improve our models further. We used the category inference system using a CNN (called CNN in the following) that was trained as in our previous study [11]. We combined our biLSTM+CFNN with CNN as follows.

Table 6 Accuracy @1/10/100 (denoted as @1, @10, and @100 in titles) and rank median of biLSTM + CFNN with and without CNN category inference.

MODEL	@1	@10	@100	median
biLSTM + CFNN	0.24	0.58	0.82	6.3
biLSTM + CFNN with CNN	0.26	0.59	0.81	5.8

Table 7 Accuracy@1/10/100 (denoted as @1, @10, and @100 in column titles) and rank median of noised and non-noised models.

MODEL	@1	@10	@100	median
biLSTM + CFNN	0.24	0.58	0.82	6.3
noised biLSTM + CFNN	0.25	0.62	0.81	5.3
noised biLSTM + CFNN with CNN	0.26	0.62	0.82	5.0
OneLook (2018)	0.34	0.55	0.76	6.0

(1) BiLSTM + CFNN searches candidate words from an input definition.
(2) The CNN predicts probabilities for each category (45 types) from the input definition.
(3) If there is a category that has an estimated probability higher than 0.95 by the CNN, choose the candidate words in the category. Otherwise, choose all the candidate words.

The accuracy@1, accuracy@10, and rank median were improved with the category inference as shown in **Table 6**.

4.3.2 Data Augmentation by Noising Definitions

Adding noise to input sentences has been shown to be a very effective data augmentation technique for some NLP tasks, such as autoencoder [12], [13] and back translation [14]. We noised training data definitions by deleting words with probability 0.1, replacing words with an <unk> token with probability 0.1, and moving one word per sentence to no further than three positions apart following a previous study [14].

Table 7 lists the results of this experiment. Noising input substantially improved accuracy @10 and the rank median of biLSTM+CFNN. Therefore, noising is also effective for the NRD. CNN in Section 4.3.1 can be combined to improve this noised biLSTM+CFNN further.

4.3.3 Pre-trained Language Representation Model for NRD

BERT [15] is a general purpose pre-trained language representation model that is based on Transformer [22] and can be fine-tuned with one additional output layer to create state-of-the-art models for a wide range of tasks.

For a reverse dictionary task, we added one fully connected layer to a pre-trained BERT-Base [*8] model. BERT-Base outputs positional embeddings and segment embeddings, then positional embeddings are added and pass through the additional fully connected layer. We dismissed segment embeddings because an NRD model requires only one sentence as its input. Because BERT-Base is a large model (even though the BERT-Base is smaller than BERT-Large), we cannot make a batch size larger than 5 in our training environment. After fine-tuning up to three epochs, this model failed to learn appropriate networks to solve

[*8] Distributed in https://github.com/google-research/bert

the reverse dictionary task (the median is over 20,000). This failure seems to have been caused by the extremely small batch size, because with such a small batch size, even our model cannot learn anything. Reverse dictionary tasks require an individual regression for each type of target word to be solved, so training the NRD on a small batch size seemingly results in local solutions.

5. Analysis

In this section, we discuss on the reason of the good performance of biLSTM + CFNN, which performed the best among the models we presented in Section 4.

5.1 Which Factor Makes biLSTM+CFNN so Accurate?

To understand what factor makes biLSTM+CFNN so accurate, we conducted further experiments under different conditions. The results are listed in **Table 8**. The following notations are used in the table.

FNN (default) : FNN introduced in Section 4.2

FNN (complex) : FNN having almost the same number of parameters as the CFNN

X **(fixed) +** *Y* : the model in which pre-trained and fixed model *X* is connected to model *Y*

As shown in Table 8, FNN (default) (1) was less accurate than the CFNN (3), but FNN (complex) (2) had almost the same accuracy as the CFNN. This means that the difference in accuracy between an FNN and the CFNN is due to just their capacity but not from their network structure. Among biLSTM+FNN (default) (5), biLSTM+FNN (complex) (6), and biLSTM+CFNN (9), only biLSTM + CFNN had higher accuracy than biLSTM (4). Both biLSTM (fixed) + FNN (complex) (8) and biLSTM (fixed) + CFNN (10) had higher accuracy than biLSTM (4), but biLSTM + CFNN (9) was more accurate than those models. BiLSTM (fixed) + FNN (default) (5) had almost the same accuracy as biLSTM (4). Therefore, we can say that biLSTM combined with a multi-layer fully connected network with sufficient capacity can complete reverse dictionary tasks. However, the entire model cannot be learned properly without the bypass structure of the multi-layer fully connected network. Separate training of biLSTM and a multi-layer fully connected network is less effective probably because training the biLSTM alone may cause overfitting. The bypass structure is essential for learning in the lower layers when lower and higher layers are simultaneously trained.

Table 8 Accuracy@1/10/100 (denoted as @1, @10, and @100 in column titles) and rank median of each models.

MODEL	@1	@10	@100	median
1. FNN (default)	0.12	0.38	0.68	28.0
2. FNN (complex)	0.16	0.45	0.70	16.0
3. CFNN	0.15	0.43	0.71	17.5
4. biLSTM	0.18	0.52	0.77	9.8
5. biLSTM + FNN (default)	0.21	0.48	0.79	13.0
6. biLSTM + FNN (complex)	0.16	0.42	0.75	17.0
7. biLSTM (fixed) + FNN (default)	0.21	0.52	0.76	10.0
8. biLSTM (fixed) + FNN (complex)	0.24	0.54	0.77	8.0
9. biLSTM + CFNN	0.24	0.58	0.82	6.3
10. biLSTM (fixed) + CFNN	0.24	0.54	0.76	7.0
11. stacked-LSTM	0.20	0.52	0.79	9.0
12. stacked-LSTM + CFNN	0.22	0.52	0.77	8.5
13. stacked-LSTM (fixed) + CFNN	0.20	0.54	0.78	8.0

Some image-processing neural networks, such as ResNet [23] and DenseNet [24], have similar bypass structures. These bypass structures are introduced to deepen learnable CNN networks. Also, ResNet behaves like ensemble learning because of its bypass structure [25]. Our bypass structure serves a somewhat different function from the others.

Stacked-LSTM (11) was more accurate than biLSTM (4), but both stacked-LSTM + CFNN (12) and stacked-LSTM (fixed) + CFNN (13) were less accurate than biLSTM + CFNN (9). The possible reason for these facts is that stacked-LSTM has too much capacity and learns what the multi-layer fully connected network can learn more effectively.

We now summarize the analysis. We can construct a neural network model to complete reverse dictionary tasks by using biLSTM followed by a multi-layer fully connected network with the bypass structure. To train this network properly, two conditions are crucial.

(1) LSTM and a multi-layer fully connected network should be trained simultaneously.

(2) LSTM should not have too much capacity.

That is, the balance between LSTM and a multi-layer fully connected network is important for constructing a better performing NRD.

Among our models, biLSTM + CFNN has the highest accuracy. This is because the capacity of biLSTM matches that of the CFNN, as shown in Table 4, and the bypass structure of the CFNN makes it possible to train the entire model at once.

We now discuss the reason of these observations in various aspects.

5.2 How does CFNN works in biLSTM + CFNN?

BiLSTM+CFNN performed substantially better than biLSTM. To understand the reason for this, we formed the following hypothesis on the function of the CFNN in biLSTM + CFNN: the CFNN draws the biLSTM-generated embedding of multiple descriptions, which are explaining same word, more similar directions.

To verify this hypothesis, we plotted the cosine similarity of the embedding at various layers of biLSTM and CFNN in **Fig. 4** (Left).

We plotted the similarity of the following combinations.

- train-train mean: A centroid of definitions and a definition in the dictionaries for the 200 words in the user descriptions were used. The results are averaged.
- train-test mean: A centroid of definitions in the dictionaries and a user description for the 200 words in the user descriptions are used. The results are averaged.
- train-train word “forget”: A centroid of definitions and a definition for the word “forget” in the dictionaries are used.
- train-train word “identify”: A centroid of definitions and a definition for the word “office” in the dictionaries are used.

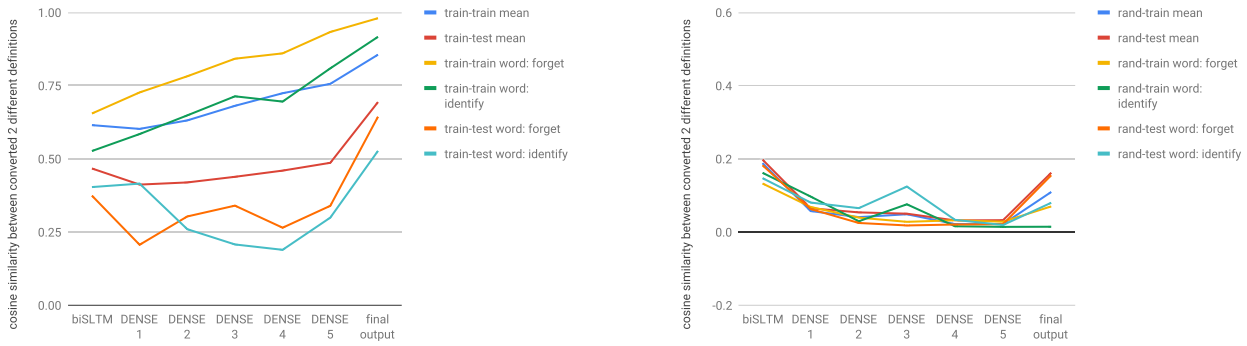


Fig. 4 Cosine similarity of the centroid of definitions and a definition (or description) of (Left) one word, (Right) different words. Vertical axis is along the cosine similarity. Horizontal axis is along the layers. n-th layer of CFNN is denoted as "DENSE n".

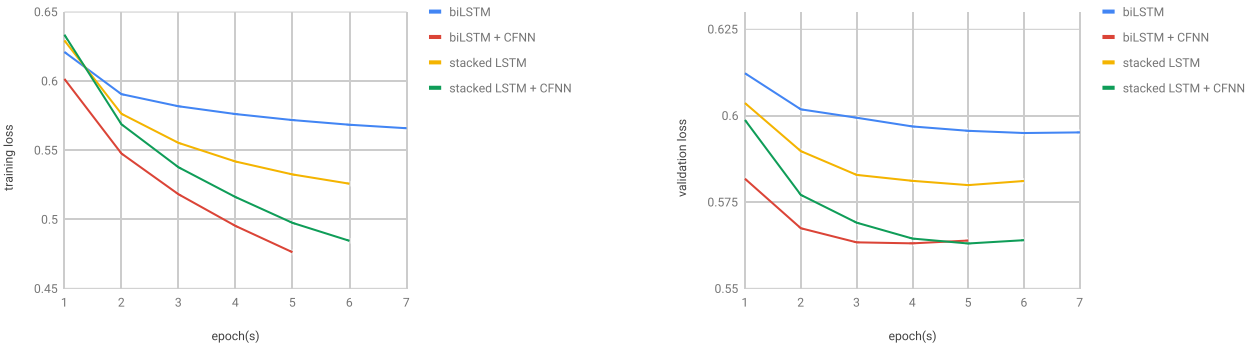


Fig. 5 (Left) Training loss for each epoch. (Right) Validation loss for each epoch. biLSTM, biLSTM + CFNN, stacked-LSTM, and stacked-LSTM + CFNN were used.

- train-test word "forget": A centroid of definitions in the dictionaries and one in user description for the word "forget" are used.
- train-test word "identify": A centroid of definitions in the dictionaries and one in user description for the word "office" are used.

In contrast, we plotted those for two different words in Fig. 4 (Right). "rand-train word forget" in this figure means a centroid of definitions for a randomly selected (and different from "forget") word in the dictionaries and a definition for the word "forget" are used, and others as well.

Comparing these figures, we can see that only definitions for one word are drawn nearer by CFNN, and definitions for different words are not. This is remarkable for the training data, but it is also seen for the test data. This result supports the hypothesis that the CFNN draws the biLSTM-generated embedding of the descriptions for one word nearer.

5.3 Why stacked-LSTM+CFNN does not perform better than biLSTM + CFNN?

Stacked-LSTM performed better than biLSTM, but stacked-LSTM+CFNN did not perform better than biLSTM + CFNN, and we found that "LSTM should not have too much capacity", as mentioned in Section 5.1. To understand this reason, we plotted the training loss and validation loss in **Fig. 5**.

From this figure:

- Connecting biLSTM before the CFNN substantially improves learning efficiency and generalization capability for the validation data much.

Table 9 Accuracy @1/10/100 (denoted as @1, @10, and @100 in titles) and rank median of biSLTM + CFNN, stacked-LSTM + CFNN, stacked-LSTM (fixed) + CFNN evaluated in the reverse dictionary task.

MODEL	@1	@10	@100	median
biLSTM + CFNN	0.24	**0.58**	**0.82**	6.3
stacked-LSTM + CFNN	0.22	0.52	0.77	8.5
stacked-LSTM (fixed) + CFNN	0.22	0.54	0.78	8.0

- Connecting stacked-LSTM before the CFNN slightly improves learning efficiency and generalization capability for the validation data.
- The generalization capability of biLSTM + CFNN and stacked-LSTM + CFNN are the same for the validation data (but not for the test data).

From these observations, we argue that there are mainly two cases for the poor performance of stacked-LSTM in stacked-LSTM+CFNN.

Case A Because stacked-LSTM has more capacity, the learning of it is slow and the training of it is incomplete.

Case B Because stacked-LSTM has more expressive power, it consumes the rules that are better learned using the CFNN and degrades overall performance. That is, stacked-LSTM learns the rules that are valid only in dictionary data (training and validation) and not in user (test) descriptions.

To test Case A, we trained stacked-LSTM first then the CFNN later. We denote this model as stacked-LSTM (fixed) + CFNN. The results of this and those for biLSTM+CFNN and stacked-LSTM+CFNN are listed in **Table 9**. Stacked-LSTM+CFNN and stacked-LSTM (fixed) + CFNN did not perform better than that

Table 10 Three test examples for reverse dictionary task evaluation. Our biLSTM+CFNN was superior for "prefer", Hill's LSTM was superior for "knowledge", OneLook Reverse Dictionary was superior for "east". OneLook Reverse Dictionary only returned top 1,000 words, so we denote rank of target word not found in returned result by OneLook Reverse Dictionary as 1,000<.

Target	Description	Model	Target Rank	Top 10 Candidates
prefer	to like one thing more than another thing	biLSTM + CFNN	**1**	**prefer**, choose, simply, either, or, preferred, preferable, certain, choosing, rather
		LSTM	106	so, even, simply, make, do, whatever, but, be, ignore, always
		OneLook	1,000<	inclination, this, inclinnation, quantity, number, odds, into, genitive, better, excess
east	one of the directions on a compass that points right when you look at it	biLSTM + CFNN	15	eastward, westward, straight, southward, northward, westwards, eastwards, north, northwards, direction
		LSTM	6,319	angle, curve, directions, diagonal, compass, angles, vertical, diagonally, angled, direction
		OneLook	**1**	**east**, point, orientation, aspect, west, projection, orientate, orient, oriented, pole
knowledge	all of the information or facts that somebody might have in their head	biLSTM + CFNN	13,569	head, heads, back, front, then, arm, hand, out, down, off
		LSTM	**47**	merely, simply, clearly, therefore, particular, necessarily, indeed, actually, rather, otherwise
		OneLook	151	know, selection, record, mind, recorded, mindfulness, credit, right, interview, diary

of biLSTM+CFNN. The pre-training of stacked-LSTM did not improve performance. Thus, because Case A does not hold, we argue that Case B, i.e., stacked-LSTM consumes the rules that are better learned using the CFNN and degrades overall performance, holds.

5.4 Qualitative Analysis

We manually inspected 200 test examples for our biLSTM + CFNN, Hill's LSTM, and OneLook Reverse Dictionary.

Compared with OneLook Reverse Dicitonary in terms of target ranks, biLSTM + CFNN was better for 87 examples, worse for 82 examples, the same for 26 examples, and unknown [*9] for 5 examples. BiLSTM + CFNN has similar performance, and seemed to work complementarily to OneLook Reverse Dictionary. From **Table 10**, biLSTM + CFNN performed better than OneLook Reverse Dictionary for the target word "prefer", and conversely for the target word "east".

Compared with LSTM in terms of target ranks, biLSTM + CFNN performed better for 138 examples, worse for 49 examples, and same for 13 examples. BiLSTM + CFNN performed mostly better than LSTM. When LSTM had better target rank than other reverse dictionaries, we found that other highly ranked words were not suitable for the context of the input description. For example, LSTM had better rank for the target word "knowledge" than biLSTM + CFNN and OneLook Reverse Dictionary as shown in Table 10, but the other top 10 candidates by LSTM were completely irrelevant to the target meaning. On the other hand, OneLook Reverse Dictionary failed to name the target word "knowledge", but the other top 10 candidates were relevant to the target meaning.

6. Conclusion

We pointed out that the previous neural network models of the NRD may have insufficient capacity; thus, we improved the performance of the NRD by using novel combinations of neural network models with sufficient capacities. We also confirmed that data augmentation by noising inputs is beneficial for the NRD, as for other NLP tasks. We applied this data augmentation and combined CNN category classification, which has been shown to be effective for the NRD, to biLSTM+CFNN (noised BiLSTM+CFNN with CNN). Noised biLSTM + CFNN with a CNN, is our best model for the NRD; thus, the NRD outperformed the OneLook Reverse Dictionary in all our metrics except accuracy @1. The performance of the NRD with our models matched that of OneLook Reverse Dictionary, which can access a large amount of resources (1,061 indexed dictionaries) and can be tuned by experts.

The LSTM structure's output adjustment by a multi-layer fully connected network with the bypass structure is more effective for reverse dictionary tasks than using complicated LSTM. This model might perform well in other NLP tasks. This investigation is for future work.

We have another future work: to investigate how the NRD bridges the gap between known dictionary definitions and unknown user descriptions. We suggested that just complicated LSTM networks does not improve the NRD performance. To reveal generalizing mechanism of the NRD will help to improve its performance.

[*9] OneLook Reverse Dictionary returns only top 1,000 words. In case of a target word found in neither of OneLook Reverse Dictionary output words nor biLSTM + CFNN output top 1,000 words, we cannot determine which target rank is higher. This is the "unknown" case.

References

[1] Schwartz, B.L. and Metcalfe, J.: Tip-of-the-tongue (TOT) states: Retrieval, behavior, and experience, *Memory & Cognition*, Vol.39, No.5, pp.737–749 (2011).

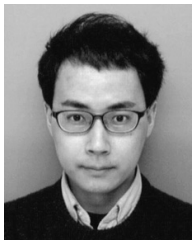
[2] Hill, F., Cho, K., Korhonen, A. and Bengio, Y.: Learning to Understand Phrases by Embedding the Dictionary, *Trans. Association for Computational Linguistics*, Vol.4, pp.17–30 (2016).

[3] Miller, G.A.: WordNet: A lexical database for English, *Comm. ACM*, Vol.38, No.11, pp.39–41 (1995).

[4] Dutoit, D. and Nugues, P.: A Lexical Database and an Algorithm to Find Words from Definitions, *Proc. 15th European Conference on Artificial Intelligence* (*ECAI2002*), pp.450–454 (2002).

[5] El-kahlout, I.D. and Oflazer, K.: Use of Wordnet for Retrieving Words from Their Meanings, *Proc. Global Wordnet Conference* (*GWC2004*),

pp.118–123 (2004).
[6] Méndez, O., Calvo, H. and Moreno-Armendáriz, M.A.: A Reverse Dictionary Based on Semantic Analysis Using WordNet, *MICAI 2013: Advances in Artificial Intelligence and Its Applications - 12th Mexican International Conference on Artificial Intelligence, Part I*, pp.275–285 (online), DOI: 10.1007/978-3-642-45114-0_22 (2013).
[7] Shaw, R., Datta, A., VanderMeer, D. and Dutta, K.: Building a scalable database-driven reverse dictionary, *IEEE Trans. Knowledge and Data Engineering*, Vol.25, No.3, pp.528–540 (2013).
[8] Thorat, S. and Choudhari, V.: Implementing a Reverse Dictionary, based on word definitions, using a Node-Graph Architecture, *Proc. COLING 2016, the 26th International Conference on Computational Linguistics*, pp.2797–2806 (2016).
[9] Bengio, Y., Ducharme, R., Vincent, P. and Jauvin, C.: A neural probabilistic language model, *Journal of Machine Learning Research*, Vol.3, No.Feb, pp.1137–1155 (2003).
[10] Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S. and Dean, J.: Distributed representations of words and phrases and their compositionality, *Advances in Neural Information Processing Systems*, pp.3111–3119 (2013).
[11] Morinaga, Y. and Yamaguchi, K.: Improvement of Reverse Dictionary by Tuning Word Vectors and Category Inference, *ICIST 2018, CCIS 920*, pp.533–545 (2018).
[12] Hill, F., Cho, K. and Korhonen, A.: Learning Distributed Representations of Sentences from Unlabelled Data, *Proc. NAACL-HLT*, pp.1367–1377 (2016).
[13] Lample, G., Conneau, A., Denoyer, L., et al.: Unsupervised Machine Translation Using Monolingual Corpora Only, *International Conference on Learning Representations* (*ICLR*) (2018).
[14] Edunov, S., Ott, M., Auli, M. and Grangier, D.: Understanding Back-Translation at Scale, *Proc. 2018 Conference on Empirical Methods in Natural Language Processing*, pp.489–500 (2018).
[15] Devlin, J., Chang, M.-W., Lee, K. and Toutanova, K.: BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, *Proc. 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1* (*Long and Short Papers*), pp.4171–4186 (2019).
[16] Hochreiter, S. and Schmidhuber, J.: Long short-term memory, *Neural Computation*, Vol.9, No.8, pp.1735–1780 (1997).
[17] Fahlman, S.E. and Lebiere, C.: The Cascade-Correlation Learning Architecture, *Advances in Neural Information Processing Systems 2*, Touretzky, D.S. (Ed.), Morgan-Kaufmann, pp.524–532 (1990) (online), available from ⟨http://papers.nips.cc/paper/207-the-cascade-correlation-learning-architecture.pdf⟩.
[18] Huang, G., Liu, Z., van der Maaten, L. and Weinberger, K.Q.: Densely Connected Convolutional Networks, *2017 IEEE Conference on Computer Vision and Pattern Recognition* (*CVPR*), pp.2261–2269 (online), DOI: 10.1109/CVPR.2017.243 (2017).
[19] Bahdanau, D., Cho, K. and Bengio, Y.: Neural machine translation by jointly learning to align and translate, arXiv preprint arXiv:1409.0473 (2014).
[20] Godin, F., Dambre, J. and De Neve, W.: Improving Language Modeling using Densely Connected Recurrent Neural Networks, *Proc. 2nd Workshop on Representation Learning for NLP*, pp.186–190 (2017).
[21] Kingma, D.P. and Ba, J.: Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980 (2014).
[22] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł. and Polosukhin, I.: Attention is all you need, *Advances in Neural Information Processing Systems*, pp.5998–6008 (2017).
[23] He, K., Zhang, X., Ren, S. and Sun, J.: Deep residual learning for image recognition, *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp.770–778 (2016).
[24] Huang, G., Liu, Z., Van Der Maaten, L. and Weinberger, K.Q.: Densely connected convolutional networks, *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp.4700–4708 (2017).
[25] Veit, A., Wilber, M.J. and Belongie, S.: Residual networks behave like ensembles of relatively shallow networks, *Advances in Neural Information Processing Systems*, pp.550–558 (2016).



Yuya Morinaga received his B.A. and M.A. degree from the University of Tokyo, in 2016 and 2019, respectively. He is a Ph.D. candidate in the University of Tokyo.

Kazunori Yamaguchi received his B.S., M.S., and Doctor of Science degrees in information science from the University of Tokyo, in 1979, 1981, and 1985, respectively. Currently, he is a professor of the University of Tokyo. His research interests are in data models and data analysis. He is a member of IPSJ.